A Fast Multilevel Implementation of Recursive Spectral Bisection for Partitioning Unstructured Problems

Stephen T. Barnard[1] and Horst D. Simon[2]
Applied Research Branch
Numerical Aerodynamic Simulation (NAS) Systems Division
NASA Ames Research Center, Mail Stop T045-1
Moffett Field, CA 94035

March 22, 1994

Abstract

Unstructured meshes are used in many large-scale scientific and engineering problems, including finite-volume methods for computational fluid dynamics and finite-element methods for structural analysis. If unstructured problems such as these are to be solved on distributed-memory parallel computers, their data structures must be partitioned and distributed across processors; if they are to be solved *efficiently*, the partitioning must maximize load balance and minimize interprocessor communication. Recently the recursive spectral bisection method (RSB) has been shown to be very effective for such partitioning problems compared to alternative methods. Unfortunately, RSB in its simplest form is rather expensive. In this report we shall describe a multilevel implementation of RSB that can attain about an order-of-magnitude improvement in run time on typical examples.

Keywords: graph partitioning, domain decomposition, MIMD machines, multilevel algorithm, spectral bisection, sparse eigenvalue problem.
AMS Subject Classification 68R10, 05C50, 65F15
CR Subject Classification G.2.2, G.1.3

[1]The author is an employee of Cray Research, Inc.

[2]The author is an employee of Computer Sciences Corporation. This work is supported through NASA Contract NAS 2-12961

1 Introduction

Unstructured meshes are used in several large-scale scientific and engineering problems, including finite-volume methods for computational fluid dynamics and finite-element methods for structural analysis. If unstructured problems such as these are to be solved on distributed-memory parallel computers, their data structures must be partitioned and distributed across processors; if they are to be solved *efficiently*, the partitioning must maximize load balance and minimize interprocessor communication. Recently, the recursive spectral bisection method (RSB) [24] has been shown to be very effective for such partitioning problems compared to alternative methods. Unfortunately, RSB in its simplest form is rather expensive. We shall describe a multilevel version of RSB that can attain about an order-of-magnitude improvement in run time on typical examples.

After its introduction in [24] RSB has found very rapid acceptance as an effective method for partitioning unstructured problems in variety of applications situations. Hammond [9] considered the implementation of an unstructured grid Euler solver on the Connection Machine 2, and found that RSB followed by cyclic pairwise exchange to be the best mapping scheme for the CM-2. Johan [11] uses RSB to partition large three dimensional finite element problems for the CM-2 and CM-5 and obtains excellent performance results. He also discusses a data-parallel implementation of RSB on the CM-2. On the Intel iPSC/860 Venkatakrishnan et al. [27] show that RSB yields the best performance results when combined with a parallel Euler solver for unstructured grids. Finally, the group at ICASE has used RSB for an efficient partitioning of three dimensional problems on the 512-processor Caltech Delta Machine [2], and is considering making RSB part of a software system for automatic partitioning and manipulation of unstructured problems. Another comparison of different partitioning methods is given by Williams [28].

There is considerable theoretical evidence that spectral bisection is in a certain sense an optimal algorithm for the graph partitioning problem. Mohar [17] has summarized earlier work and given some bounds on the edge counts for spectral bisection. Pothen et al. [23] derived additional bounds, and Hendrickson and Leland [10] generalized this work by considering higher eigenvectors and more general partitioning problems. In spite of these good theorectical foundations, it has not yet been completely established that RSB is indeed the best possible algorithm in all practical situations. Alternatives are Farhat's greedy algorithm [4, 5], the geometric approach by Teng [26] and by Miller et al. [16], spectral partitionings based on higher eigenvectors resulting in tetra- or octasection as suggested in [10], or methods such as simulated annealing [18] (for recent detailed study see the thesis by Mansour [14]). A more detailed comparison of these and possible other candidate algorithms on a common set of realistic large problems from CFD and structural mechanics is planned by the authors and Farhat in the near future.

The implementation of RSB requires the computation of the smallest nontrivial eigenvector of the Laplacian matrix associated with the graph of the problem. The Laplacian matrix is a sparse, symmetric positive semidefinite matrix and of the same order as the problem. In the previous implementation [24] this vector has been computed with the unfactored Lanczos algorithm. Here we propose a considerably faster algorithm for computing the eigenvector based on a combination of some powerful numerical techniques: a multilevel algorithm, and Rayleigh quotient iteration [21, 22] combined with SYMMLQ [19] for the eigenvector extraction.

The use of a multilevel algorithm for the computation of an eigenvector of a general sparse matrix is new computational technique in its own right, independent of the partitioning issue. In this paper we have combined in a unique way a number of numerical techniques. Multgrid methods for eigenvalue problems have been discussed previously by McCormick and his collaborators [15, 1, 13], but not for unstructured grids. Inner-outer iterations for eigenvalue problems are discussed in the literature, some examples are the work by Lewis [12] and Szyld [25]. Generally these inner-outer iterations have not been as widely accepted as a method for eigenvalue computation as the Lanczos algorithm. By merging the two approaches we can take advantage of the best features of both: the multigrid scheme provides quickly a coarse approximation to the desired eigenvector, and the inner-outer iteration scheme converges quickly, once provided with a reasonable starting vector.

We also propose here a new scheme for coarsening an existing unstructured general sparse problem. Our algorithm is based on the concept of maximal independent sets. Related, but different approaches are "sparsification", i.e. the removal of edges from the graph [3], or "graph contraction", i.e. the merging of vertices [14]. What both approaches have in common with the coarsening proposed here, is that some fundamental graph property will be inherited by the smaller (i.e. smaller number of vertices or edges or both) graph, and can be computed faster, since only a smaller dataset has to be considered.

The paper is organized as follows: Section 2 defines the partitioning problem and reviews the RSB method. We then introduce the multilevel approach to RSB in Section 3 and describe the multilevel method in detail. The following issues are addressed: contracting the grid by selecting successive maximal independent sets of vertices and connecting these vertices into a smaller graph, methods for finding the eigenvectors efficiently, and interpolating eigenvectors from smaller to finer graphs. The performance of the multilevel method is compared to the original single-level method in Section 4. Finally, we draw some conclusions and discuss our plans to apply the multilevel RSB method to dynamic repartitioning of adaptive meshes.

2 Partitioning

Unstructured meshes are represented as undirected graphs using sparse-matrix data structures. The algorithms that operate on these meshes typically involve the repeated application of the same basic computation at each vertex of the graph. For simplicity, therefore, we assume that a unit amount of work is associated with each vertex.[3] Before a problem using an unstructured mesh can be solved on a parallel, distributed-memory computer, the graph must be divided into subgraphs associated with each processor. Furthermore, these subgraphs must be of very nearly equal sizes (i.e., equal numbers of vertices) to achieve a good load balance, and the number of edges connecting the subgraphs must be minimized to avoid excessive communication between processors.

Given an undirected graph $G = (V, E)$, the partitioning problem is to find a set of disjoint subsets of V, $P = \{V_i\}$, satisfying certain conditions related to *load balance* and *communication*. To optimize load balance one must minimize the variance of the sizes of the vertex subsets: $\sum_i (V_i{}^2 - \overline{V}^2)$. To optimize interprocessor communication one must minimize the sizes of the *cut sets* of edges: $\sum_{i,j} |C_{i,j}|$, where $C_{i,j} = \{(v, w) \mid v \in V_i, w \in V_j\}$. De-

[3]This assumption may not be strictly valid; for example, vertices on the boundary of a fluid-dynamics mesh may require significantly more work. The partitioning techniques to be discussed can be usually extended to such situations in a straightforward way.

pending on the topology of the interprocessor communication network, the cost of communication is also affected by the distance that the data must travel (i.e., the number of wires that must be traversed). However, modern parallel computers employ cut-through routing, which makes the cost of communication nearly independent of the distance between communicating processors. Because of this fact, and because there are many varieties of network topologies, we have chosen to ignore this component of the communication cost. Numerical studies on the iPSC/860 which demonstrate that this is a reasonable assumption are reported in [27].

The recursive spectral bisection method approaches the partitioning problem with a graph bisection strategy developed by Pothen, Simon, and Liou [23]. The method is based on computing the eigenvector corresponding to the second smallest eigenvalue of the Laplacian matrix $L(G) = (l_{i,j})$ of the graph G, defined as:

$$l_{i,j} = \begin{cases} -1 & \text{if } (v_i, v_j) \in E \\ \deg(v_i) & \text{if } i = j \\ 0 & \text{otherwise.} \end{cases}$$

It is easily seen that $L(G) = D - A$, where A is the adjacency matrix of the graph, and D is the diagonal matrix of vertex degrees. Traditionally, spectral properties of the adjacency matrix have been investigated in graph theory. However, Mohar [17] has gathered convincing evidence that the Laplacian matrix is the more natural object for the study of spectral properties of graphs. One of the reasons is that $L(G)$ is closely related to the Laplacian operator. Specifically, if we consider the standard discrete five point Laplacian on a rectangular grid, then the discrete Laplacian and the Laplacian matrix coincide, if Neumann boundary conditions are imposed. Thus one can consider the Laplacian matrix as a generalization of the discrete Laplacian operator for general graphs. This relationship is explored in more detail in [23].

The Laplacian matrix has a number of intriguing properties, which are just listed here. For details and proof see [17]. First note that the bilinear form associated with the Laplacian matrix can be written as follows:

$$x^t L x = \sum_{(v,w) \in E} (x_v - x_w)^2.$$

From this it follows that Laplacian matrix is positive semidefinite and its smallest eigenvalue is zero, with an associated eigenvector of all ones. If G is connected then the second largest eigenvalue λ_2 is positive. The eigenvector x_2 associated with λ_2 (called the *Fiedler vector*) contains important directional information about the graph (see [6, 7, 8]): the components of x_2 are weights on the corresponding vertices of G such that differences of the components provide information about the distances between the vertices. Sorting the vertices by the components of the Fiedler vector, therefore, effectively orders the graph in a way that keeps vertices "close" to one another.

Another way of deriving a matrix formulation of the graph bisection problem is as follows. Let us consider only the first partition, and let E_c denote the set of edges cut by the first partition, i.e. $|E_c| = \{e|e \in E; e = (v_1, v_2); v_1 \in V_1; v_2 \in V_2\}$. Let $n = |V|$ be even and define a *partition vector* to be a vector p with components equal to $+1$ or -1, i.e. $p = (+1, -1, -1, +1, \ldots\ldots -1, -1)^T$, with $e^T p = 0$. Here $e = (1, 1, \ldots.1)^T$. The orthogonality condition $e^T p = 0$ assures that there is an equal number of positive and negative components in p. Denote the set of all partition vectors $\mathcal{P}$. The key observation is then

$$4E_c = ||Lp||_1, \tag{1}$$

where $||x||_1 = \sum_{i=1}^{n} |x_i|$. We can construct the partition induced by the vector p, by considering the two sets of vertices V_1 and V_2, corresponding to where entries in p are $+1$ and where entries are -1. Then (1) follows from the fact that for vertices in V_1 which have no neighbors in V_2 (and vice versa) the corresponding entry in Lp is zero, since the entries in each row of the Laplacian matrix sum to zero. Only entries of Lp corresponding to vertices with neighbors in the other set are nonzero, and a simple counting arguments yields (1).

Then one can show by minimizing over all (suitably normalized) vectors in $\mathcal{R}^n$:

$$\begin{aligned} \min_{p \in \mathcal{P}, p^T e = 0} E_c &= \min_{p \in \mathcal{P}, p^T e = 0} \frac{1}{4} ||Lp||_1 \\ &\geq \min_{p \in \mathcal{R}^n, ||p||_2 = \sqrt{n}, p^T e = 0} \frac{1}{4} ||Lp||_1 \\ &\geq \frac{1}{4} \sqrt{n} |\lambda_2| \end{aligned} \tag{2}$$

Equation (2) shows that the second eigenvector of the Laplacian matrix minimizes a closely related continuous problem. We have thus replaced the solution of an intractable discrete problem, the graph bisection problem, by the solution of a well understood linear algebra problem, the computation of an eigenvector. In the spectral bisection algorithm we then proceed to chose a partition based on the partition vector closest to the second eigenvector. In general this yields a good (but not necessarily the best) partition.

The original implementation of RSB in [24] used a modified Lanczos algorithm to compute the Fiedler vectors. It applied this computation recursively as follows:

1. Compute the Fiedler vector for the graph using the unfactored Lanczos algorithm.

2. Sort the vertices according to the sizes of the components of the Fiedler vector.

3. Assign half the vertices to each subdomain.

4. Repeat recursively to each subdomain until the desired number of partitions is obtained.

The computation that dominates the cost of this algorithm is the first step, the computation of the Fiedler vector.

In the next section we describe a multilevel technique for computing the Fiedler vector much more efficiently. The basic idea is closely related to multigrid methods (although it differs from conventional multigrid methods in some respects). A series of successively smaller matrices is constructed, each of which approximates its larger predecessor. At some point the matrix is so small that the Lanczos algorithm can compute the Fiedler vector of the Laplacian matrix in a negligible amount of time. This vector is then interpolated into the next higher level, producing a high grade approximation for the Fiedler vector for the Laplacian matrix corresponding to this finer level. The interpolated vector is then improved using the Rayleigh Quotient Iteration algorithm. The process continues until the Fiedler vector for the original matrix is obtained.

3 Multilevel Partitioning

The multilevel RSB method requires three elements to be added to the basic single-level RSB algorithm:

- **Contraction**: We must be able to construct a series of smaller graphs that in some sense retain the global structure of the original large graph.
- **Interpolation**: Given a Fiedler vector of a contracted graph, we must be able to interpolate this vector to the next larger graph in a way that provides a good approximation to next Fiedler vector.
- **Refinement**: Given an approximate Fiedler vector for a graph, we must be able to compute a more accurate vector efficiently.

3.1 Contraction

The first step in contracting the graph $G = (V, E)$ is to select a subset of vertices, $V' \subset V$, that are evenly distributed over G. We choose V' to be a *maximal independent set* with respect to G. V' is an independent set with respect to G if for all $v \in V'$ $(v, w) \in E \Rightarrow w \notin V'$. V' is a *maximal* independent set if the addition of any vertex to V' would make it no longer an independent set. A maximal independent set can be found very efficiently with a simple greedy algorithm.

Once V' has been found we construct the contracted graph $G' = (V', E')$ as illustrated in Figure 1.

The solid-colored vertices in Figure 1 are those chosen as the vertices of the contracted graph. (Note that these vertices do not comprise a maximal independent set. The construction of the contracted graph can work with any subset of vertices. A relatively sparse set illustrates the method more clearly.[4]) The basic idea is to grow domains in G around vertices in V', adding an edge to E' whenever two domains intersect. A hashing scheme is

[4]Originally, we selected vertices by simply choosing a set at random, with some specified probability of selecting any vertex. The maximal independent set works somewhat better because the vertices in V' are guaranteed to be distributed evenly over G. A future parallel implementation may use the random-selection method, however, because finding a maximal independent set efficiently in parallel is a difficult problem.

Figure 1: Constructing a contracted graph.

used to avoid adding redundant edges. The algorithm terminates after all vertices in V have been visited.

One significant difference between this method and a standard multigrid approach is that we construct dynamically a new series of contracted graphs for every subproblem rather than constructing a single series of contracted graphs to be used thoughout the algorithm. A single series *could* be used in the following way: When a partition is subdivided the vertices of its two components could be projected onto the appropriate contracted graph and the algorithm could proceed with the associated subgraphs. Unfortunately, a component may become disconnected, but the subgraph it projects to could be connected. In such a case the Fiedler vector would be incorrect. By contracting every subproblem independently we ensure that the connectivity of partitions is maintained.

3.2 Interpolation

Suppose we have found a Fiedler vector $\mathbf{f}' =< f'_i >$, $i = 1 \ldots n'$ of a contracted graph G', where $n' = |V'|$. An interpolation operation constructs an expansion of this vector, $\mathbf{f}^0 =< f^0{}_i >$, $i = 1 \ldots n$, that can be used as an approximation of the Fiedler vector G. Interpolation consists of two steps: injection and averaging.

We use information gathered in the contraction operation — namely, a mapping from vertices in V' to vertices in V. Let this mapping be $m(i)$, $i = 1 \ldots n'$ such that $m(i)$ is the index of the vertex in V from which the ith vertex in V' was derived during the computation of the maximal independent set. The injection operation first "seeds" the vector $\mathbf{f}^0$ by placing components of $\mathbf{f}'$ into $\mathbf{f}^0$:

$$f^0{}_{m(i)} = f'{}_i, \; i = 1 \ldots n' \, .$$

The remaining components of $\mathbf{f}^0$ are set by averaging the components of their neighbors that have been set by injection Since V' is a maximal independent set with respect to V we can be assured that these components will have such neighbors.

3.3 Refinement

The Lanczos algorithm works well in the absence of prior information about the eigenvector, but in our situation it does not effectively take advantage of a good initial approximation. Therefore, although we still use the Lanczos algorithm as described in [20] to find the Fiedler vector for the smallest graphs, we use Rayleigh quotient iteration (RQI) [21, 22] to refine the approximate Fiedler vector for the higher levels. We start RQI with the interpolated vector as described above, and use its Rayleigh quotient as an initial guess for the eigenvalue. Since we have already a very good approximation to the Fiedler vector and our accuracy requirements are modest, RQI takes usually only a few steps to convergence. One of the features of the SYMMLQ implementation is that for very ill-conditioned matrices, SYMMLQ terminates early, when a good approximate eigenvector is found. This early termination of SYMMLQ causes more RQI steps to be made then one would expect, becuase of the cubic convergence of RQI. However, the total number of matrix vector operations made in all RQI iterations combined was usually less than ten. Thus the RQI/SYMMLQ combination proved to be very efficient for refining the coarse grid approximation for an eigenvector.

The Rayleigh quotient iteration algorithm is shown in Figure 2. This algorithm requires that a linear system be solved on each iteration. Since we do not want to employ any factorization scheme, we use SYMMLQ [19],

```
function rqi($\mathbf{v}, L$)
$\theta \leftarrow \mathbf{v}^T L\mathbf{v}$
do
    solve $(L - \theta I)\mathbf{x} = \mathbf{v}$
    $\mathbf{v} \leftarrow \mathbf{x}/||\mathbf{x}||$
    $\theta \leftarrow \mathbf{v}^T L\mathbf{v}$
    $\rho \leftarrow \sqrt{(L\mathbf{v})^T(L\mathbf{v}) - \theta^2}$
until $\rho < \epsilon$
return $\mathbf{v}$
```

Figure 2: Rayleigh quotient iteration

which is an extension of the conjugate gradient algorithm designed to work for indefinite matrices.

The three steps discussed above — contraction, interpolation, and refinement – are put together in a recursive, multilevel routine to find the Fiedler vector of a graph, shown in Figure 3. Note that it is not necessary to actually construct the Laplacian matrix. Instead, we have modified a sparse matrix-vector multiplication routine to use the Laplacian matrix implicitly defined by the adjacency information.

4 Performance

Tables 1, 2, 3, and 4 compare the performance of the multilevel partitioning algorithm (ML) to the original single-level algorithm (SL) for problems of various sizes. Hammond is a rather small two-dimensional CFD mesh, Barth5 is a somewhat larger mesh of the same type, and PWT (which stands for Pressured Wind Tunnel) is a large finite-element mesh with a toroidal topology, and IN3C is a very large three-dimensional finite-element mesh.[5] The cutoff for graph contraction (n in Figure 3) is 100 for the Hammond, Barth5, and

[5]The Hammond, Barth5, and PWT problems were run on a Silicon Graphics workstation with one 20 MHZ IP6 processor and 16 Mbytes of main memory. The IN3C problem was run on a Silicon Graphics workstation with eight 33 MHZ IP7 processors and 256 Mbytes of main memory.

```
function Fiedler($G$)
$L \leftarrow$ Laplacian matrix of $G$
if $|V| > n$ then
    $G' \leftarrow$ contract($G$)
    $\mathbf{v}' \leftarrow$ Fiedler($G'$)
    $\mathbf{v} \leftarrow$ interpolate($\mathbf{v}', L$)
    $\mathbf{v} \leftarrow$ rqi($\mathbf{v}, L$)
else
    $\mathbf{v} \leftarrow$ Lanczos($L$)
endif
return $\mathbf{v}$
```

Figure 3: Recursive procedure to find the Fiedler vector.

	SL		ML		
partitions	time (sec)	edges cut	time (sec)	edges cut	speedup
2	40.4	117	3.97	117	10.2
4	74.7	258	7.14	274	10.5
8	95.7	466	11.5	481	8.3
16	108	758	16.0	770	6.8
32	117	1215	20.4	1199	5.7
64	124	1893	26.2	1863	4.7
128	129	2788	31.5	2798	4.1

Table 1: Performance on the Hammond mesh ($|V| = 4720$, $|E| = 13722$).

	SL		ML		
partitions	time (sec)	edges cut	time (sec)	edges cut	speedup
2	182	181	11.3	164	16.0
4	411	470	21.2	492	19.4
8	534	813	30.7	808	17.4
16	617	1330	41.2	1375	15.0
32	670	2048	53.4	2131	12.5
64	704	3190	68.8	3252	10.2
128	734	4966	88.0	4925	8.3

Table 2: Performance on the Barth5 mesh ($|V| = 15606$, $|E| = 45878$).

	SL		ML		
partitions	time (sec)	edges cut	time (sec)	edges cut	speedup
2	893	423	28.0	376	31.9
4	1533	829	48.1	942	31.9
8	1847	1558	72.4	1684	25.5
16	2055	3091	106	3302	19.4
32	2284	5973	162	6273	14.1
64	2452	9478	203	9790	12.1
128	2559	13222	249	13535	10.3

Table 3: Performance on the PWT mesh ($|V| = 36519$, $|E| = 144794$).

	SL		ML		
partitions	time (sec)	edges cut	time (sec)	edges cut	speedup
2	2766	2455	152	2366	18.2
4	4985	6383	250	5639	20.0
8	6811	11760	350	10086	19.5
16	8138	18988	461	16955	17.7
32	9141	26696	591	25388	15.5
64	9907	39536	750	36876	13.2
128	10508	56754	933	52732	11.3

Table 4: Performance on the IN3C mesh ($|V| = 262620$, $|E| = 764268$).

PWT problems, and 500 for the IN3C problem.

The quality of the partitions, as measured by the sizes of the sizes of the cut sets, is nearly the same for both algorithms. For 128 partitions the ML result is 0.4% worse than the SL result for Hammond, 0.8% better for Barth5, and 2.4% worse for PWT, and 7.1% better for IN3C.

The performance advantage of the multilevel algorithm over the single level algorithm is dependent on the size of the problem. The speedup for computing 128 partitions ranges from 4.1 for the smallest problem to 11.3 for the largest. This effect is clearly illustrated in Figures 4, 5, and 6.

These graphs show the average run times of the ML and SL algorithms for each subproblem. For the Hammond mesh (Figure 4) the smallest subproblems are smaller than the cutoff for contracting the graph. Therefore, the Lanczos algorithm is used exclusively to compute the Fiedler vector so SL and ML take the same amount of time (except for a small additional overhead in ML). For the larger subproblems, however, the performance advantage of ML improves because the multilevel structure is "deeper" and the execution time of Lanczos algorithm at the coarsest level represents a smaller proportion of the total time.

Figure 4: Average performance for each subproblem size: Hammond mesh. Each point indicates the average execution time for solving each subproblem.

Figure 5: Average performance for each subproblem size: Barth5 mesh.

Figure 6: Average performance for each subproblem size: Pressured Wind Tunnel.

5 Conclusions and Future Work

In this paper we have combined a number of computational techniques ranging from multilevel methods to discrete graph algorithms to improve the performance of the recursive spectral bisection algorithm. We have demonstrated that the multilevel approach yields substantially better performance with little or no loss of quality in the computed partition. On large problems we have seen an order of magnitude reduction in the execution time on workstations. Furthermore, we believe that the combination of multilevel techniqes with RQI and SYMMLQ may be useful in general for computing eigenvectors of large sparse matrices.

The techniques of grid coarsening and eigenvector interpolation are of potential use in adapting the RSB algorithm for dynamically changing grids, such as in problems with adaptive meshes, and/or local grid refinement. We intent to apply the ideas presented here to the dynamic repartitioning in the near future.

Acknowledgements. We would like to thank Niel Madsen for providing us with the large test problem, Michael Saunders, Stanford University, for giving us permission to use his SYMMLQ code, Robert Schreiber, RIACS at NASA Ames Research Center, for suggesting to use maximal independent sets, and John Gilbert (Xerox PARC), Bruce Hendrickson (Sandia), Rob Leland (Sandia), and V. Venkatakrishnan (NASA Ames) for stimulating and hepful discussions about the topic of this paper.

References

[1] A. Brandt, S. McCormick, and J. Ruge. Multigrid methods for differential eigenproblems. *SIAM J. Sci. Stat. Comp.*, 4(2):244 – 260, 1983.

[2] R. Das, D.J. Mavriplis, J. Saltz, S. Gupta, and R. Ponnusamy. The design and implementation of parallel unstructured Euler solver using software primitives. In *AIAA 30th Aerospace Sciences Meeting*, 1992. Paper AIAA-92-0562.

[3] D. Eppstein, Z. Galil, G. Italiano, and A. Nissenzweig. Sparsification - a technique for speeding up dynamic graph algorithms. In *Proceedings of FOCS '92*, 1992.

[4] C. Farhat. A simple and efficient automatic FEM domain decomposer. *Computers and Structures*, 28(5):579 – 602, 1988.

[5] C. Farhat. On the mapping of massively parallel processors onto finite element graphs. *Computers and Structures*, 32(2):347 – 353, 1989.

[6] M. Fiedler. Algebraic connectivity of graphs. *Czechoslovak Math. J.*, 23(98):298 – 305, 1973.

[7] M. Fiedler. Eigenvectors of acyclic matrices. *Czechoslovak Math. J.*, 25(100):607 – 618, 1975.

[8] M. Fiedler. A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory. *Czechoslovak Math. J.*, 25(100):619 – 633, 1975.

[9] S. Hammond. *Mapping Unstructured Grid Computations to Massively Parallel Computers*. PhD thesis, RPI, June 1992. RIACS Report 92.14.

[10] B. Hendrickson and R. Leland. An improved spectral graph partitioning algorithm for mapping parallel computations. Technical Report SAND92-1460, UC - 405, Sandia Natl. Lab., Albuquerque, N.M., September 1992.

[11] Z. Johan. *Data Parallel Finite Element Techniques for Large-Scale Computational Fluid Dynamics*. PhD thesis, Stanford University, July 1992.

[12] J.G. Lewis. *Algorithms for Sparse Matrix Eigenvalue Problems*. PhD thesis, Stanford University, Dept. of Computer Science, 1977.

[13] J. Mandel and S. McCormick. A multilevel variational method for $Au = \lambda Bu$ on composite grids. *Journal of Comp. Physics*, 80(2):442 – 452, 1989.

[14] Nashat Mansour. *Physical Optimization Algorithms for Mapping Data to Distributed-Memory Multiprocessors*. PhD thesis, Dept. of Computer Science, Syracuse University, Syracuse, NY, August 1992.

[15] Stephen F. McCormick. A mesh refinement method for $Ax = \lambda Bx$. *Mathematics of Computation*, 36(154):485 – 498, 1981.

[16] Gary L. Miller, Shang-Hua Teng, William Thurston, and Stephen Vavasis. Automatic mesh partitioning. School of Computer Science, Carnegie Mellon Univ., Pittsburgh, PA, 1992.

[17] B. Mohar. *The Laplacian Spectrum of Graphs*, pages 871–898. J. Wiley, New York, 1991.

[18] B. Nour-Omid, A. Raefsky, and G. Lyzenga. Solving finite element equations on concurrent computers. In A. K. Noor, editor, *Parallel Computations and their Impact on Mechanics*, pages 209 – 227, New York, 1986. American Soc. of Mech. Eng.

[19] C. C. Paige and M. A. Saunders. Solution of sparse indefinite systems of linear equations. *SIAM J. Num. Anal.*, 12:617 – 629, 1974.

[20] B. Parlett, H. Simon, and L. Stringer. Estimating the largest eigenvalue with the Lanczos algorithm. *Math. Comp.*, 38:153 – 165, 1982.

[21] B. N. Parlett. The Rayleigh quotient iteration and some generalizations for nonnormal matrices. *Math. Comp.*, 28(127):679 – 693, 1974.

[22] B.N. Parlett. *The Symmetric Eigenvalue Problem.* Prentice Hall, Englewood Cliffs, New Jersey, 1980.

[23] A. Pothen, H. Simon, and K.-P. Liou. Partitioning sparse matrices with eigenvectors of graphs. *SIAM J. Mat. Anal. Appl.*, 11(3):430 – 452, 1990.

[24] H. D. Simon. Partitioning of unstructured problems for parallel processsing. *Computing Systems in Engineering*, 2(2/3):135 – 148, 1991.

[25] Daniel B. Szyld. Criteria for combining inverse and rayleigh quotient iteration. *SIAM Journal on Numerical Analysis*, 25:1369 – 1375, 1988.

[26] Shang-Hua Teng. *Points, Spheres, and Separators, A Unified Geometric Approach to Graph Partitioning.* PhD thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, August 1991.

[27] V. Venkatakrishnan, H. Simon, and T. Barth. A mimd implementation of a parallel euler solver for unstructured grids. *The Journal of Supercomputing*, 6(2):117 – 127, 1992.

[28] R. D. Williams. Performance of dynamic load balancing algorithms for unstructured mesh calculations. Technical Report C3P913, California Institute of Technology, Pasadena, California, June 1990.